Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 12 (2014) 3 – 10

Procedia Technology

The 7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013)

Inference of probabilistic grammars in different rules systems of natural languages

László Kovács[a]*, Zsolt Tóth[a].

[a] *University of Miskolc, Miskolc-Egyetemváros, H 3515, Hungary*

Abstract

Grammar induction is an important and current area within computational linguistics. The results in induction of string transformation rules can be applied not only in linguistics but in many other pattern recognition tasks. The task of grammar induction belongs in general to NP -hard problems. The paper analyses the main rule types in natural languages and in string transformation systems. The paper focuses on the inference of inflectional rule systems which differs in many aspects from the traditional production rule system of Chomsky grammars. The work compares the main candidate methods on the learning of objective-case in the Hungarian language.

Keywords: grammar induction; string transformation; rule classification; intersection lattice;

1. Introduction

A set of sequences containing elements of a given type can be considered as a formal language. The atomic symbols of the language, i.e. the building elements of the sentences are called terminal symbols of the language. In the practice, the symbols may correspond to any arbitrary objects like words of a human language or the nucleotides of a DNA chain. The grammar of a language describes the production rules to decompose a sequence into the

* Corresponding author. Tel.: +36-20.3319765.
E-mail address: kovacs@iit.uni-miskolc.hu

doi:10.1016/j.protcy.2013.12.449

terminal symbols. The grammar of a language can be used to describe the structure of the language. The grammar induction focuses on the discovery of the most compact grammar for a given set of training sequences.

The task of grammar induction is generally an NP-hard problem. The simplest case in grammar induction is the induction of regular grammars. A formal grammar is defined a $G=<T,N,P,S>$ where T is the set of terminal symbols, N is the set of non-terminal symbols, S is the sentence symbols where $S \in N$. The P component corresponds to the set of productions rules. For regular grammar, the P contains rules of type $A \rightarrow b$ and $A \rightarrow Bb$ where A and B are non-terminal symbols and b is a terminal symbol. For practical cases, the regular grammar is usually a good model of the real grammar but in most complex cases like in human languages, a probabilistic context free grammar (PCFG) should be used. In PCFG, every production rule is assigned to a probability value and a rule has the form $A' \rightarrow B'$ where A' and B' are sequences of terminal and non-terminals symbols and A' must contain at least one non-terminal symbol. Based on the probability values of the rules, the probability distribution of the valid sentences can be generated too. The main benefit of the PCFG approach is that this class of grammar does not require negative examples during the training, only positive examples are enough to construct a grammar.

2. Rule categories in natural languages

The base foundation of grammar analysis is the theory of formal grammars. The grammar system defined in formal grammars is based on the Chomsky approach. This means that the rules are production rules how to construct the symbol sequences from the given alphabet. Thus the main concepts of the grammar are the segmentation and concatenation operators. This grammar describes the rules how to build up the valid sequences of the language from atomic unites. This rule system is appropriate to analyse the ordering of the building blocks of the sentences. In many human languages this ordering is the main method to encode the semantic. On the other hand, many languages contain additional formalism for representation of the semantic. Thus a key problem in the computational linguistics is how appropriate is the Chomsky grammar system in the modeling of human languages.

The simplest approach is that regular grammar is powerful enough to describe human languages. This approach is based on the fact that every finite language can be described with regular grammar and every human language is also finite. The main drawback of this approach is the huge complexity of the corresponding regular grammar. The first regular grammar model for natural language was developed by Hockett [18] in 1955. Sullivan [19] also argued for regularity where its reasoning is based on the finite structure of our brain. The brain can be modelled with a finite automaton. The number of states of the resulting non-deterministic finite automaton is approximated with value $10^{1000000000}$. Thus the only way to prove that NL are more complex than regular grammars is to exhibit some infinite sequences of grammatical sentences. Chomsky argued that the sentence structure of human languages shows such infinite length pattern which requires some kind of memory too. The best known example is the sentence:

"*A white male (whom a white male)^n (hired)^n hired another white male.*"

There are also very strong arguments that NLs are not context-free. In the area of formal languages in conjunction with computational linguistics and natural language processing the formulation of the notion of mildly context-sensitive languages has been appeared and used. These classes are proper subclasses of the class of context-sensitive languages and proper superclasses of the class of context-free languages [20].

Beside word ordering, the other main formal operation for encoding semantic is the word inflection [12]. In the agglutinative languages a word may have many different forms depending on the semantic role. For example in the bask language, a noun may have several hundred versions. The tested Hungarian language also belongs to this family of languages. In this language there are about 30 cases which can be combined with each other. The accusative is a frequently used case in word inflection:

Látom a házat [←ház] (I see the house).

The stem form of the house is in Hungarian the word 'ház'. This word is altered to the form 'házat' to denote the role of accusative case. The main difficulty in this grammar is that the postfix tag can vary depending from the stem

form. In the Hungarian language there are about 80 different forms of accusative. Some typical examples are shown in the Table 1.

Table 1. Inflection examples.

Stem	Accusative case
kefe	*kefét*
szó	*szót*
labda	*labdát*
dob	*dobot*
álom	*álmot*

The formalism of inflection rules is very different from the traditional Chomsky grammars system. Here, a string transformation rule should be described. In the literature, there are only few approaches presented on the inference of inflectional rule system.

The morphology of inflectional languages is usually described with paradigm formalism. A paradigm [17] is a table describing the stem and the inflectional forms of an example word, representing a given inflectional class (see Table 1). This approach is called WP (Word and Paradigm) model. The corresponding paradigm set is constructed manually by linguistic experts. The proposed Murf framework in [17] is a program intended to induce a morphological transducer from traditional style paradigm sets. The transducer produces the forms in the paradigms and generalises common features, reducing redundancy in the paradigms.

The inflection given by the paradigm table requires different learning methods than the induction of Chomsky grammars [11]. The goal of inflection rule is the matching of stem form (lemma) with the appropriate derivation rule. As the number of different derivation rules is finite, the rules are called derivation classes. In this approach, the inflection of derivation classes can be considered as a classification task.

3. Inference of probabilistic CFG

The grammar induction methods were developed originally for Chomsky grammars. For induction of PCFG, the most widely used methods are the probabilistic inductive CYK (PCYK), the TBL algorithm [1] and the ADIOS method. The test results and the theoretical analysis show that the TBL algorithm has a very high cost value; for a single sentence the construction of the grammar table is in $O(N^3)$. For a set of training sentences, the first phase generates a primitive grammar for every sample sentence. In the second phase, the primitive grammars are integrated with a heuristic method. One solution is the application of a genetic algorithm to determine the overlapping non-terminal symbol definitions. The applied GA method usually causes an additional high increase in the computational costs. The ADIOS method uses a frequent pattern distillation phase to determine the appropriate non-terminal symbols. The method generates a sequence graph from the training sequences, where every word corresponds to a single node of the graph. The PCYK algorithm uses an efficient parse table structure to discover the significant production rules. In the parse table, every non-terminal symbol is assigned to a frequency value, denoting the probability of deduction of a sentence segment from the given non-terminal. The calculation of these probability values can be implemented with a bottom-up algorithm.

ADIOS [6] performs automatic distillation of structures. It is an unsupervised Probabilistic Context Free Grammar induction algorithm. The algorithm extracts the frequent patterns from the training corpus to determine the production rules. The sentences of the training set are considered as unique paths in a pseudograph [7]. The graph is directed and has multiple edges. Its nodes are the words of the training set and there are two special nodes called “begin” and “end”.

Each path starts with the begin node and ends with the end node thus the graph is coherent. To extract patterns, it uses the Motif Extension algorithm. The patterns are the frequent sub-paths of the graph. After the extraction of the selected frequent sub-path, it becomes a new node which replaces the original path and the graph will be rewritten. The graph grows and gets a new node in every iteration. Moreover, a similarity relationship can be defined among the extracted patterns. For example, the " want you to" and "want her to" paths have very similar contents. In the generalization phase, these patterns will be merged such as "want X to" and X = {you; her}.

We recently focus on the representation and the pattern acquisition of the ADIOS algorithm. The comparison of the TBL [5,4], Improved TBL [7,1] and Inductive CYK [2,3] algorithms showed that the selected representation has a great impact on the inference method. Based on the analysis [8], the iterative algorithms yields result faster than non-iterative algorithms. The analysis stresses the importance of the Motif Extraction method. This algorithm returns the most frequent sub-path in the given pseudo graph. The algorithm examines each possible sub-path in the graph and determines PR, PL, DR, DL values for each sub-paths. It uses an exhaustive searching method which yields the best solution however it is very costly. For example, if the training set consists of n sentences and each sentence is m-length, then there could be $m(m+1)/2$ sub-path for each sentence and $nm(m+1)/2$ sub-paths total. Thus the number of the possible sub-paths grows as $O(nm^2)$. It means the Motif Extraction algorithm itself has a huge time-cost in each iteration. The above mentioned values are the following

$$P_R(e_1,e_n)=\frac{\left|e_1\to e_2\to\ldots\to e_n\right|}{\left|e_1\to e_2\to\ldots\to e_{n-1}\right|}$$

$$P_L(e_1,e_n)=\frac{\left|e_1\to e_2\to\ldots\to e_n\right|}{\left|e_2\to e_2\to\ldots\to e_n\right|}$$

$$D_R(e_1,e_n)=\frac{P_R(e_1,e_n)}{P_R(e_1,e_{n-1})}$$

$$D_L(e_1,e_n)=\frac{P_R(e_1,e_n)}{P_R(e_2,e_n)}.$$

The graph representation of the corpus plays an important role in the pattern acquisition. It gives the theoretical background of the search and pattern extraction. From the point of view of efficiency, also the implementation plays a key role. The graph is directed, contains multiple edges and the paths have to be encoded too for the Motif Extraction algorithm. Our implementation of the graph is a list of paths and a path is a sequence of words. The Motif Extraction algorithm works on this structure and rewrites the sentences in every iteration

4. Inference of inflectional grammar

The inflection [10] is usually considered as a classification problem and some kind of classification method is applied in the implementation systems. A special characteristic of inflection system is the high complexity and irregularity. In [9], a neural network architecture was used to cope with this complexity. The proposed network was trained with 3,244 Serbian nouns to predict inflected phonological forms from a specification of a word's lemma, gender, number, and case, and generalized to untrained cases. The proposed feed-forward network consists of four layers. In the third layer a recurrent architecture was implemented to enable a sequential output. Considering the input layer, every lemma used in the language has a specific unit. In the test implementation, 407 lemma nodes were used. Beside lemma nodes, the input layer consists also units for the investigated cases too. After activation of the actual lemma and case units, the network outputs the inflected form of the lemma. The main drawback of this architecture is that every lemma requires a specific unit, the set of lemmas should be fixed apriori.

Our investigation aims at developing a more flexible structure. Two models was developed and compared, the first is based on the neural network approach while the second one uses an intersection lattice structure. The input of the grammar learning system is a list of lemma – inflected word pairs.

$$L=\left\{(s_i,s'_i)\right\}.$$

In the first, preprocessing phase, the inflection rules are recognized. The applied rules correspond to substitution rules, the application of this substitution operation generates from lemma s, the corresponding inflected form is s'. Every discovered rule is considered as a transformation class r_j. The training set for the second phase is the set of lemmas and corresponding transformation class labels:

$$T = \{(s_i, r_j)\}.$$

The first investigated approach to learn the classification function was the Counter Propagation Network[16, 17]. This network consists of three layers, where the hidden layer corresponds to a Kohonen's SOM layer, while the output layer is a Grossberg competitive layer.

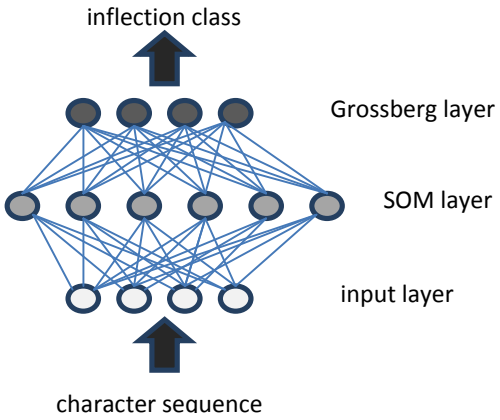

Fig. 1. The structure of CPN neural network.

The nodes of the hidden layers represent cluster centers in the object space. The network learns the optimal positions of the cluster centers. Regions with high number of objects have a higher density of cluster nodes. The third layer performs a nearest neighbor classification, where the class labels of the neighboring training samples determine the class of the tested input object. A key aspect of network structure is the interpretation of the input nodes. Usually, the input nodes correspond to the attributes of the test objects. In our implementation, the input nodes are assigned to the different letters at the different positions of the lemma word. Thus the total number of input nodes is equal to $n*m$ where n is the size of alphabet and m is the threshold length of words.

Another key parameter is the number of nodes in the hidden layer. If this value is too low, the neighborhoods of the cluster centers are inhomogeneous. The estimation of the network size is a weak point of this method.

Another investigated approach is the application of an intersection lattice. This model is based on the analogy with the concept lattice architecture. In the theory of formal concept analysis (FCA) [13, 14] , the input information is summarized in a context matrix K. The rows of the matrix correspond to the objects and the columns denote the attributes. The cell value denotes whether the given attribute is valid at the object or not.

For a given context matrix, a set of formal concepts can be generated. A formal concept C is a pair of object set A and attribute set B, where

$$A = \{a \mid \forall b \in B : (a,b) \in K\}$$

$$B = \{b \mid \forall a \in A : (a,b) \in K\}.$$

Within the set of formal concepts from K, a sub-concept relationship can be defined where

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 .$$

Based on this relationship, a lattice of concepts can be constructed to a given K context.

The relationship in this lattice corresponds to the generalization relationship of common objects [15]. The rows of the context matrix are usually atomic concepts, while the derived concepts are generated with the intersection of the existing and of the incoming concepts. Extending the attribute set with a class attribute, the lattice can be used as a classification method [14]. In this extended lattice, the maximal consistent concepts are those concepts which are consistent (every descendant concepts belong to the same class label) and no of its ancestor classes is consistent. The testing process localizes that maximal consistent concept in the lattice whose attribute set is a subset of the input object's attribute set. The class label of the found maximal consistent concept will be assigned to the input object

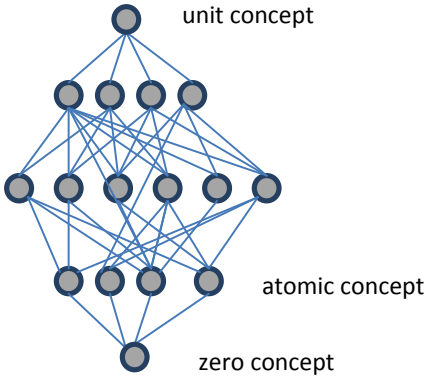

Fig. 2. The structure of concept lattice

In the implemented system, the objects of the context correspond to elements of the training set

$$T = \{(s_i, r_j)\}.$$

The characters of the lemma components are used as object attributes. The intension part of the concepts is here a character sequence instead of character set. The intersection operation yields the common string sequences of the two operand strings. The definition of common substring may have different interpretation and it strongly influences the efficiency of the classification process. The main steps of the intersection lattice method are the followings:

- generate the list of new intersection concepts
- eliminate the redundancy in the list
- determine the nearest container concepts for the new concepts
- determine the nearest contained objects for the new concepts
- insert the new concepts into the lattice
- adjust the consistency indicator of the concepts
- extract from the lattice the maximal consistent elements.

The cost of lattice management depends significantly on the number of the concept nodes. The traditional lattice construction methods usually belong to the *O(N3)* complexity class, where *N* denotes the number of nodes is the lattice. Unfortunately, having a context with *M* rows, the number of generated concepts may have in $O(2^M)$. Thus, the lattice oriented approach requires an appropriate reduction method to eliminate the unimportant nodes from the lattice. In our current investigation the postfix inflection system was tested, thus the end of the lemmas has larger importance than the head part

5. Test results

META is a novel framework for grammar induction methods [9][8]. Its aim is to provide a common environment for grammar induction and parsing methods thus the different methods can be compared and analysed jointly. It provides classes to represent formal grammars and interfaces to manipulate them. The grammar description is

general so any kind of grammar from Chomsky hierarchy can be represented in it. There are two kind of rule types for grammars the Production Rule and Probabilistic Rule. The Grammar class is generic thus a Grammar object can be probabilistic or non-probabilistic. There are generic interfaces Parse Strategy and Learn Strategy to implement own parsing and learning algorithms. Thus META is an extensible framework to manage formal grammars, and it provides a way to implement and test different algorithms.

Regarding the inflection problem, a training set containing 2600 training pairs was selected for test experiments. The training set was constructed from 2600 different noun words and the inflection for accusative case was used. The items in the input file contain a pair of words. The first word is the lemma word and the second word is the inflected word.

In the tests, two main measures were investigated, the accuracy and the execution time. In the tests, the described CPN and intersection lattice methods were compared. The experimental measurements have a large agreement with the theoretical considerations. The intersection lattice structure shows a polynomial execution time function (Fig. 4), thus for larger problem domains, a more efficient reduction component should be invented.

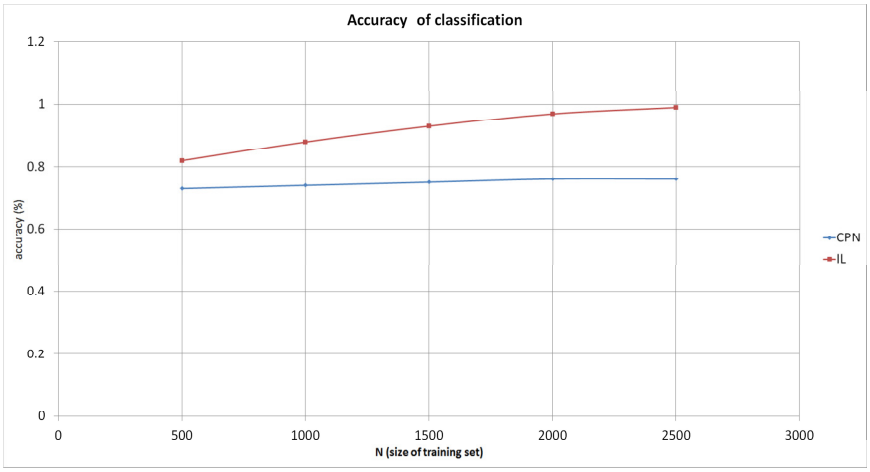

Fig. 3. The accuracy comparison of the CPN and IL methods

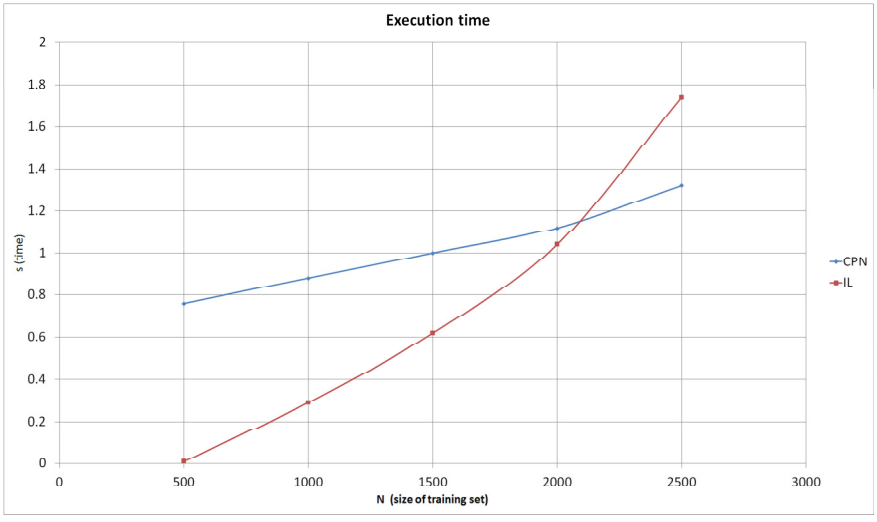

Fig. 4. The time cost comparison of the CPN and IL methods

Regarding the accuracy, the intersection lattice method dominates the CPN neural network structure (Fig 3.). The main benefit of the IL method is that it provides a better generalization mechanism for untrained cases than the CPN method.

6. Conclusion

The grammar theory in computational linguistics is dominated by the traditional Chomsky formal grammars. This theory focuses on the constructional rules of sentences from base building blocks. These rules determine the symbol set of the components and describe the ordering of components. In many natural language exists a different grammar component, namely the inflection of the words. The traditional Chomsky rule system is not very efficient on this problem; usually some type of classification method is applied for word inflection. The presented paper introduces and investigates two main methods for this task. The first method uses on a traditional neural network structure, while the other is based on the formal concept theory.

The paper presents the base version of the proposed architectures and verifies the applicability of these architectures. As the theoretical considerations and the experimental results show, both versions are suitable for the learning the inflection rules. The CPN neural network has linear cost function, thus it is suitable for large problems too. The drawback of this approach is the relative low classification accuracy, as it is a hard problem to find an appropriate distance function which is in agreement with the inflectional rules. The IL method provides a higher accuracy and it can manage the exceptional cases too. The drawback of this method is the higher execution cost for larger context matrices. The goal of further investigation is to optimize the learning algorithms of the presented alternative methods before to apply them for larger corpus with 300000 training word pairs.

References

[1] Jaworski M, Unold O. Improved TBL algorithm for learning context-free grammar, Proc. of International Multi-conference on ISSN, 2007; 1896: 7094.
[2] Nakamura K, Ishiwata T. Synthesizing context free grammars from sample strings based on inductive CYK algorithm. Grammatical Inference: Algorithms and Applications, 2000: 20-32.
[3] Nakamura K, Matsumoto M. Incremental learning of context free grammars. Grammatical Inference: Algorithms and Applications, 2002: 749-753.
[4] Sakakibara Y. Learning context-free grammars using tabular representations. Pattern Recognition, 38(9), (2005) pp.1372-1383
[5] Sakakibara Y, Kondo M. GA-based learning of context-free grammars using tabular representations. Proc. of Machine learning Int. Workshop, 1999: 354-360.
[6] Solan Z, Horn D., Ruppin E, Edelman S. Unsupervised learning of natural languages. Proc. of the National Academy of Sciences of the United States of America, 2005; 102(33): 11629.
[7] Solan Z., Ruppin E, Horn D, Edelman S. Automatic acquisition and e_cient representation of syntactic structures. Advances in Neural Information Processing Systems, 2003: 107-116.
[8] Tóth Zs, Kovács L. CFG extension for meta framework. Proc. of Intelligent Engineering Systems (INES), 2012: 495-500.
[9] Mirkovic J, Seidenberg M., Joanisse M. Rules versus statistics: Insights from a highly inflected language, Cognitive Science, 2010; 35: 638-681.
[10] Liebhaber M. A computer model of inflection learning, Behavior Research Methods, Instruments & Computers, 1988;20(2): 246-254.
[11] Wilson C. Learning phonology with substantive bias: an experimental and computational study of velar palatalization, Cognitive Science, 1988; 30: 945-982.
[12] Cermak M, Horacek P, Meduna A. Rule-restricted automaton grammar transducers: power and linguistic applications, Mathematics for Application, 2012; 1: 13-35.
[13] Godin R, Missaoui R, Alaoiu H. Incremental concept formation algorithms based on Galois lattices, Computational Intelligence, 1995; 11(2): 246-267.
[14] Zhao Y, Yao Y. Classification based on logical concept analysis, Proc. of Canadian Society for Computational Studies of Intelligence, 2006: 419-430.
[15] Valtchev P, Missaoui R. Building concept lattices from parts: generalizing the incremental method, Proc. of ICCS01, 2001: 290-303.
[16] Hecht-Nielsen R. Counterpropagation Networks, Applied Optics, 1987; 26(23): 4979-4984.
[17] Carlson L. Inducing a morphological transducer from inflectional paradigms, Inquires into Words, Constraints and Contexts, Antti Arppe (eds.), 2005: 18-24.
[18] Hockett C. A manual of phonology, Waverly Press, 1955.
[19] Sullivan W. Syntax and linguistic semantics in stratificational theory, Current Approaches to Syntax, 1980: 321-327.
[20] Winter S. Formal language theory for natural language processing, Proc. of ACL, 2002: 71-76